

CLAIMS

1 1. A method for executing a program written for an original computer
2 system on a different host computer system, comprising the steps of:
3 fetching program code;
4 translating program code;
5 emitting translated program code into at least one code cache; and
6 executing translated code within the at least one code cache in lieu of
7 associated program code when a semantic function of the associated program code is
8 requested.

1 2. The method of claim 1, wherein the step of fetching program code
2 comprises fetching program instructions with an emulator.

1 3. The method of claim 2, wherein the emulator is an
2 interpreter/emulator.

1 4. The method of claim 1, wherein the step of translating the program
2 code comprises translating program instructions with a just-in-time compiler.

1 5. The method of claim 1, wherein the step of emitting translated program
2 code into at least one code cache comprises emitting translated program code into the
3 at least one code cache via an application programming interface.

1 6. The method of claim 1, further comprising the step of interpreting and
2 executing program code that has not be emitted into the at least one code cache.

1 7. The method of claim 6, further comprising the step of emulating
2 actions that would have been performed by the original computer system during
3 execution.

1 8. The method of claim 1, further comprising the step of, prior to emitting
2 translated program code, growing a code fragment by linking program instructions
3 together.

1 9. The method of claim 8, wherein the step of linking program
2 instructions together comprises linking program instructions together with a just-in-
3 time compiler.

1 10. A system for executing program code on a host computer system,
2 comprising:
3 means for translating program code;
4 means for emitting translated program code into at least one code cache; and
5 means for executing translated code within the at least one code cache in lieu
6 of associated program code when a semantic function of the associated program code
7 is requested.

1 11. The system of claim 10, wherein the means for translating the program
2 code comprise a just-in-time compiler.

1 12. The system of claim 10, wherein the means for emitting translated
2 program code into at least one code cache comprise an application programming
3 interface.

1 13. The system of claim 10, further comprising means for interpreting and
2 executing program code.

1 14. The system of claim 10, further comprising means for emulating
2 actions that would have been performed during execution by an original computer
3 system for which the program code was written.

1 15. An emulation program configured to emulate an original computer
2 system for which a program was written, the emulation program stored on a
3 computer-readable medium and comprising:

4 logic configured to translate program code;

5 logic configured to emit code fragment translations of program code into at
6 least one code cache; and

7 logic configured to execute the code fragments within the at least one code
8 cache in lieu of associated program code when a semantic function of the associated
9 program code is requested.

1 16. The program of claim 15, wherein the logic configured to translate the
2 program code comprises a just-in-time compiler.

1 17. The program of claim 15, wherein the logic configured to emit code
2 fragment translations comprises an application programming interface.

1 18. The program of claim 15, further comprising logic configured to
2 interpret and execute program code.

1 19. The program of claim 15, further comprising logic configured to
2 emulate actions of the original computer system for which the program was written.

1 20. A system for executing program code that was written for an original
2 computer system on a different host computer system, comprising:

3 an emulator;

4 a translator;

5 a virtual machine that comprises a dynamic execution layer interface including
6 a core having at least one code cache in which code fragments can be cached and
7 executed; and

8 an application programming interface that links the translator to the virtual
9 machine.

1 21. The system of claim 20, wherein the emulator comprises an
2 interpreter/emulator.

1 22. The system of claim 20, wherein the translator comprises a just-in-time
2 compiler.

1 23. The system of claim 20, wherein the application programming
2 interface comprises a set of functions available to the translator including an emit
3 fragment function with which the translator can emit code fragments into the at least
4 one code cache and an execute function with which the translator can request
5 execution of code fragments contained within the at least one code cache.

1 24. The system of claim 23, wherein the application programming
2 interface is configured such that the emit fragment function can be used to perform at
3 least one of tracking a cached code fragment and associating metadata with a cached
4 code fragment.

1 25. The system of claim 23, wherein the set of application programming
2 interface functions comprises a lookup fragment function with which cached code
3 fragments can be retrieved.

1 26. The system of claim 23, wherein the set of application programming
2 interface functions comprises an invalidate fragment function with which individual
3 cached code fragments can be invalidated.

1 27. The system of claim 23, wherein the set of application programming
2 interface functions comprises a enumerate fragment function with which cached code
3 fragments can be enumerated using metadata associated with the fragments.

1 28. The system of claim 23, wherein the set of application programming
2 interface functions comprises a cache flush function in which code caches of the
3 dynamic execution layer interface can be flushed.

1 29. The system of claim 23, wherein the set of application programming
2 interface functions comprises an install callback function with which the translator
3 can be notified as to events that occur within the dynamic execution layer interface.

1 30. An application programming interface configured to link a translator to
2 a dynamic execution layer interface in an computer system emulating system,
3 comprising:

4 a set of functions available to the translator including:

5 an emit fragment function with which the translator can emit code
6 fragments into code caches of the dynamic execution layer interface, and

7 an execute function with which the translator can request execution of
8 code fragments contained within the at least one code cache.

1 31. The application programming interface of claim 30, wherein the emit
2 fragment function can be used to perform at least one of tracking a cached code
3 fragment and associating metadata with a cached code fragment.

1 32. The application programming interface of claim 30, wherein the set of
2 functions further comprises a lookup fragment function with which cached code
3 fragments can be retrieved.

1 33. The application programming interface of claim 30, wherein the set of
2 functions comprises an invalidate fragment function with which individual cached
3 code fragments can be invalidated.

1 34. The application programming interface of claim 30, wherein the set of
2 functions comprises a enumerate fragment function with which cached code
3 fragments can be enumerated using metadata associated with the fragments.

1 35. The application programming interface of claim 30, wherein the set of
2 functions comprises a cache flush function in which code caches of the dynamic
3 execution layer interface can be flushed.

1 36. The application programming interface of claim 30, wherein the set of
2 functions comprises an install callback function with which the translator can be
3 notified as to particular events that occur within the dynamic execution layer
4 interface.